VBA: Create Address Grid Lines From Polygons

Contributed by Bert Granberg
06, Jan. 2010

In some areas, the rural address grid is created by dividing Public Land Survey System (PLSS) sections by a fixed number. As sections are approximately 1 mile by 1 mile, if you want an address grid that has 10 blocks to the mile, 10 block lines for each direction would need to be created for each section to create an address grid reference map.

The code below will create the block or grid lines from a set of polygons that carry attributes for the number of divisions in each direction, the starting address for each direction, and the quadrant that the polygon is in.

For the code to work as written, the polygons will need to carrying the appropriate attributes in these fieldnames:
- QUAD: the grid quadrant each polygon is in (NW, NE, SW, or SE)
- DIVX: The numer of divisions to be made across the polygon in the EW direction (ex. 0 for origin, 800, 1600, etc)
- DIVY: The number of divisions to be made for the polygon in the NS direction (ex. 0 for origin, 800, 1600, etc)
- LOWX: The lowest EW grid address to be assigned in the EW direction (ex. 0 for origin, 800, 1600, etc)
- LOWX: The lowest NS grid address to be assigned in the NS direction (ex. 0 for origin, 800, 1600, etc)

Here is an example of input and output:

Input polygons with attribute values shown as labels

Output grid lines (minimal labeling shown)

```
Public Sub buildAddressGridLinesFromPolygons()

    Dim outPath As String
    Dim polygonLayerIndex As Long
    '**************************
    '***** SET THESE PARAMETERS

    'the file geodatabase where the new address grid line feature class will be created
    outPath = "c:/UDOTLRS/LRSBert.gdb"
    polygonLayerIndex = 1

    '***** END PARAMETERS
    '**************************

    Dim pMxDoc As IMxDocument
    Dim pMap As IMap
    Dim pPolyLayer As IFeatureLayer
    Dim pPolyFC As IFeatureClass
    Dim pOutWS As IFeatureWorkspace
    Dim pOutFields As IFields
    Dim pOutFC As IFeatureClass
    Dim pGDS As IGeoDataset
    Dim pOutSR As ISpatialReference
    Dim datestamp As String
    Dim pOutFeature As IFeature
    Dim pPolyline As IPolyline
    Dim pPolygon As IPolygon
    Dim c As Long
```

```
Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
Set pPolyLayer = pMap.Layer(polygonLayerIndex)
Set pPolyFC = pPolyLayer.FeatureClass
Set pGDS = pPolyFC
Set pOutSR = pGDS.SpatialReference

datestamp = Format(Now, "yyyymmddhhmmss")


Set pOutWS = openFGDBWS(outPath)
Set pOutFields = createFields(esriGeometryPolyline, pOutSR, False)
Set pOutFC = createFeatureClass(pOutWS, "AddLines" & datestamp, esriFTSimple, esriGeometryPolyline, pOutFields)

Dim pPolyFCursor As IFeatureCursor
Dim pPolyFeature As IFeature
Set pPolyFCursor = pPolyFC.Search(Nothing, True)
Set pPolyFeature = pPolyFCursor.NextFeature

Do Until pPolyFeature Is Nothing
   c = c + 1
   Debug.Print c
   Set pPolygon = pPolyFeature.Shape


   Dim pPC As IPointCollection
   Dim pPoint As IPoint
   Dim pEnv As IEnvelope
   Dim tempPt As IPoint
   Dim v As Long
   Dim pPO As IProximityOperator
   Set pPC = pPolygon
   Set pPO = pPolygon
   Set pEnv = pPolygon.Envelope
   Set tempPt = New esriGeometry.Point

   Dim pNWPt As IPoint
   Dim pNEPt As IPoint
   Dim pSEPt As IPoint
   Dim pSWPt As IPoint

   'Find NW Corner
   tempPt.PutCoords pEnv.XMin, pEnv.YMax
   Set pNWPt = pPO.ReturnNearestPoint(tempPt, esriNoExtension)
   'Find NE Corner
   tempPt.PutCoords pEnv.XMax, pEnv.YMax
   Set pNEPt = pPO.ReturnNearestPoint(tempPt, esriNoExtension)
   'Find SE Corner
   tempPt.PutCoords pEnv.XMax, pEnv.YMin
   Set pSEPt = pPO.ReturnNearestPoint(tempPt, esriNoExtension)
   'Find SW Corner
   tempPt.PutCoords pEnv.XMin, pEnv.YMin
   Set pSWPt = pPO.ReturnNearestPoint(tempPt, esriNoExtension)

   Dim xDivs As Long
   Dim xStartVal As Long
   Dim yDivs As Long
   Dim yStartVal As Long

   Dim EQuad As Boolean
   Dim NQuad As Boolean
   Dim startFromPt As IPoint
   Dim startToPt As IPoint
   Dim endFromPt As IPoint
```

```
Dim endToPt As IPoint
Dim startLine As ILine
Dim endLine As ILine
Dim quadStr As String
Set startLine = New Line
Set endLine = New Line

quadStr = pPolyFeature.value(pPolyFC.FindField("QUAD"))

EQuad = False
If Right(quadStr, 1) = "E" Then
   EQuad = True
End If

NQuad = False
If Left(quadStr, 1) = "N" Then
   NQuad = True
End If


xDivs = pPolyFeature.value(pPolyFC.FindField("DIV_X"))
yDivs = pPolyFeature.value(pPolyFC.FindField("DIV_Y"))
xStartVal = pPolyFeature.value(pPolyFC.FindField("LOW_X"))
yStartVal = pPolyFeature.value(pPolyFC.FindField("LOW_Y"))
'NE
If EQuad And NQuad Then

   'Do Horizontal lines
   Set startFromPt = pSWPt
   Set startToPt = pNWPt
   Set endFromPt = pSEPt
   Set endToPt = pNEPt
   startLine.PutCoords startFromPt, startToPt
   endLine.PutCoords endFromPt, endToPt
   If xDivs > 0 Then
      Call writelines(pOutFC, pPO, startLine, endLine, yDivs, yStartVal, "N")
   End If

   'now do vertical lines
   Set startFromPt = pSWPt
   Set startToPt = pSEPt
   Set endFromPt = pNWPt
   Set endToPt = pNEPt
   startLine.PutCoords startFromPt, startToPt
   endLine.PutCoords endFromPt, endToPt
   If xDivs > 0 Then
      Call writelines(pOutFC, pPO, startLine, endLine, xDivs, xStartVal, "E")
   End If
'SE
ElseIf EQuad And (Not NQuad) Then

   'Do Horizontal lines
   Set startFromPt = pNWPt
   Set startToPt = pSWPt
   Set endFromPt = pNEPt
   Set endToPt = pSEPt
   startLine.PutCoords startFromPt, startToPt
   endLine.PutCoords endFromPt, endToPt
   If yDivs > 0 Then
      Call writelines(pOutFC, pPO, startLine, endLine, yDivs, yStartVal, "S")
   End If

   'now do vertical lines
   Set startFromPt = pNWPt
   Set startToPt = pNEPt
```

```
        Set endFromPt = pSWPt
        Set endToPt = pSEPt
        startLine.PutCoords startFromPt, startToPt
        endLine.PutCoords endFromPt, endToPt
        If xDivs > 0 Then
            Call writelines(pOutFC, pPO, startLine, endLine, xDivs, xStartVal, "E")
        End If

    'NW
    ElseIf (Not EQuad) And NQuad Then

        'Do Horizontal lines
        Set startFromPt = pSEPt
        Set startToPt = pNEPt
        Set endFromPt = pSWPt
        Set endToPt = pNWPt
        startLine.PutCoords startFromPt, startToPt
        endLine.PutCoords endFromPt, endToPt
        If yDivs > 0 Then
            Call writelines(pOutFC, pPO, startLine, endLine, yDivs, yStartVal, "N")
        End If

        'now do vertical lines
        Set startFromPt = pSEPt
        Set startToPt = pSWPt
        Set endFromPt = pNEPt
        Set endToPt = pNWPt
        startLine.PutCoords startFromPt, startToPt
        endLine.PutCoords endFromPt, endToPt
        If xDivs > 0 Then
            Call writelines(pOutFC, pPO, startLine, endLine, xDivs, xStartVal, "W")
        End If



    'SW
    ElseIf (Not EQuad) And (Not NQuad) Then

        'Do Horizontal lines
        Set startFromPt = pNEPt
        Set startToPt = pSEPt
        Set endFromPt = pNWPt
        Set endToPt = pSWPt
        startLine.PutCoords startFromPt, startToPt
        endLine.PutCoords endFromPt, endToPt
        If yDivs > 0 Then
            Call writelines(pOutFC, pPO, startLine, endLine, yDivs, yStartVal, "S")
        End If

        'now do vertical lines
        Set startFromPt = pNEPt
        Set startToPt = pNWPt
        Set endFromPt = pSEPt
        Set endToPt = pSWPt
        startLine.PutCoords startFromPt, startToPt
        endLine.PutCoords endFromPt, endToPt
        If xDivs > 0 Then
            Call writelines(pOutFC, pPO, startLine, endLine, xDivs, xStartVal, "W")
        End If

    End If
    Set pPolyFeature = pPolyFCursor.NextFeature
  Loop
End Sub
```

```
Public Sub writelines(inFc As IFeatureClass, inPolygonPO As IProximityOperator, startLine As ICurve, endLine As
ICurve, xDivs As Long, xStartVal As Long, sufdir As String)

    Dim d As Long
    Dim sName As Long
    Dim pStartPoint As IPoint
    Dim pEndPoint As IPoint
    Dim pLine As ILine
    Dim pGC As IGeometryCollection
    Dim pSC As ISegmentCollection
    Dim pOutF As IFeature
    Set pStartPoint = New Point
    Set pEndPoint = New Point

    For d = 0 To xDivs - 1
        Set pOutF = inFc.CreateFeature
        startLine.QueryPoint esriNoExtension, (d / xDivs), True, pStartPoint
        endLine.QueryPoint esriNoExtension, (d / xDivs), True, pEndPoint

        'now adjust point to position that is actually on polygon
        Set pStartPoint = inPolygonPO.ReturnNearestPoint(pStartPoint, esriNoExtension)
        Set pStartPoint = inPolygonPO.ReturnNearestPoint(pStartPoint, esriNoExtension)

        Set pLine = New Line
        pLine.PutCoords pStartPoint, pEndPoint
        Set pSC = New Path
        pSC.AddSegment pLine
        Set pGC = New Polyline
        pGC.AddGeometry pSC
        Set pOutF.Shape = pGC
        pOutF.value(2) = xStartVal + d * 100
        pOutF.value(3) = sufdir
        pOutF.Store
    Next d
End Sub


Public Function createFeatureClass(featWorkspace As IFeatureWorkspace, _
                                   Name As String, _
                                   featType As esriFeatureType, _
                                   geomType As esriGeometryType, _
                                   pFields As IFields _
                                   ) As IFeatureClass

  On Error GoTo EH

  Set createFeatureClass = Nothing
  If featWorkspace Is Nothing Then Exit Function
  If Name = "" Then Exit Function

  Dim pCLSID As UID
  Set pCLSID = Nothing
  Set pCLSID = New UID

    '' determine the appropriate geometry type corresponding the the feature type
  Select Case featType
      Case esriFTSimple
        pCLSID.value = "esricore.Feature"
        If geomType = esriGeometryLine Then geomType = esriGeometryPolyline
      Case esriFTSimpleJunction
        geomType = esriGeometryPoint
        pCLSID.value = "esricore.SimpleJunctionFeature"
      Case esriFTComplexJunction
```

```
      pCLSID.value = "esricore.ComplexJunctionFeature"
    Case esriFTSimpleEdge
      geomType = esriGeometryPolyline
      pCLSID.value = "esricore.SimpleEdgeFeature"
    Case esriFTComplexEdge
      geomType = esriGeometryPolyline
      pCLSID.value = "esricore.ComplexEdgeFeature"
    Case esriFTAnnotation
      Exit Function
  End Select


  ' establish the class extension
  Dim pCLSEXT As UID
  Set pCLSEXT = Nothing

  ' locate the shape field
  Dim strShapeFld As String
  Dim j As Integer
  For j = 0 To pFields.FieldCount - 1
    If pFields.Field(j).Type = esriFieldTypeGeometry Then
      strShapeFld = pFields.Field(j).Name
    End If
  Next

  Set createFeatureClass = featWorkspace.createFeatureClass(Name, pFields, pCLSID, _
                           pCLSEXT, featType, strShapeFld, "")

  Exit Function
EH:
    MsgBox Err.Description, vbInformation, "createWorkspaceFeatureClass"
End Function


Public Function createFields(geomType As Long, pSR As ISpatialReference, _
                             hasM As Boolean) As IFields
    Dim pField As IField
    Dim pFields As IFields
    Dim pFieldEdit As IFieldEdit
    Dim pFieldsEdit As IFieldsEdit
    Dim hasmcoord As Boolean


    'Create new Fields collection
    Set pFields = New Fields
    Set pFieldsEdit = pFields
    'pFieldsEdit.FieldCount = 1

    ''
    '' create the geometry field
    ''
    Dim pGeomDef As IGeometryDef
    Set pGeomDef = New GeometryDef
    Dim pGeomDefEdit As IGeometryDefEdit
    Set pGeomDefEdit = pGeomDef

    ' assign the spatial reference
    'Dim pSR As ISpatialReference
    If pSR Is Nothing Then
        Set pSR = New UnknownCoordinateSystem
        pSR.SetFalseOriginAndUnits 0, 0, 100
    End If

    pSR.SetMFalseOriginAndUnits -100000, 1000
```

```
    If Not hasM Then
       hasmcoord = False
    Else
       hasmcoord = True
    End If

    '' assign the geometry definiton properties.
    With pGeomDefEdit
      .GeometryType = geomType
      .GridCount = 1
      .GridSize(0) = 560000
      .AvgNumPoints = 200
      .hasM = hasmcoord
      .HasZ = False
      Set .SpatialReference = pSR
    End With

    Set pField = New Field
    Set pFieldEdit = pField

    pFieldEdit.Name = "Shape"
    pFieldEdit.Type = esriFieldTypeGeometry
    Set pFieldEdit.GeometryDef = pGeomDef
    pFieldsEdit.AddField pField


    'Create Object ID Field
    Set pField = New Field
    Set pFieldEdit = pField

    With pFieldEdit
       .Name = "OBJECTID"
       .AliasName = "FID"
       .Type = esriFieldTypeOID
    End With
    pFieldsEdit.AddField pField

    Set pField = New Field
    Set pFieldEdit = pField
    With pFieldEdit
       .Length = 10
       .Name = "S_NAME"
       .Type = esriFieldTypeString
    End With
    pFieldsEdit.AddField pField

    Set pField = New Field
    Set pFieldEdit = pField
    With pFieldEdit
       .Length = 1
       .Name = "SUF_DIR"
       .Type = esriFieldTypeString
    End With
    pFieldsEdit.AddField pField


    Set pFields = pFieldsEdit

    Set createFields = pFields

End Function

Public Function openFGDBWS(inPath As String) As IFeatureWorkspace
       Dim pFGDBWSFactory As IWorkspaceFactory
       Set pFGDBWSFactory = New esriDataSourcesGDB.FileGDBWorkspaceFactory
```

```
    Set openFGDBWS = pFGDBWSFactory.OpenFromFile(inPath, 0)
End Function
```